# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/735,678 | 12/16/2003 | Mark Justin Moore | 60707-1250(GV181) | 3580 |

57286 7590 06/09/2009
THOMAS, KAYDEN, HORSTEMEYER & RISLEY, L.L.P.
600 Galleria Parkway, Suite 1500
ATLANTA, GA 30339-5948

| EXAMINER | |
|---|---|
| ARCOS, CAROLINE H | |
| **ART UNIT** | **PAPER NUMBER** |
| 2195 | |
| **MAIL DATE** | **DELIVERY MODE** |
| 06/09/2009 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| | | | |
|---|---|---|---|
| ***Office Action Summary*** | **Application No.**<br>10/735,678 | **Applicant(s)**<br>MOORE ET AL. | |
| | **Examiner**<br>CAROLINE ARCOS | **Art Unit**<br>2195 | |

***-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --***

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *16 February 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-3,5-14 and 16-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,5-14 and 16-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *16 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

## DETAILED ACTION

1. Claims 1-3, 5-14, and 16-22 are pending for examination.

Applicant is reminded that amendments to the application must be made with respect to the official electronic file, and not any other version of the application, such as the published version of the application.

### *Specification*

2. Applicant is reminded of the proper content of an abstract of the disclosure.

A patent abstract is a concise statement of the technical disclosure of the patent and should include that which is new in the art to which the invention pertains. If the patent is of a basic nature, the entire technical disclosure may be new in the art, and the abstract should be directed to the entire disclosure. If the patent is in the nature of an improvement in an old apparatus, process, product, or composition, the abstract should include the technical disclosure of the improvement. In certain patents, particularly those for compounds and compositions, wherein the process for making and/or the use thereof are not obvious, the abstract should set forth a process for making and/or use thereof. If the new technical disclosure involves modifications or alternatives, the abstract should mention by way of example the preferred modification or alternative.

The abstract should not refer to purported merits or speculative applications of the invention and should not compare the invention with the prior art.

Where applicable, the abstract should include the following:
(1) if a machine or apparatus, its organization and operation;
(2) if an article, its method of making;
(3) if a chemical compound, its identity and use;
(4) if a mixture, its ingredients;
(5) if a process, the steps.

Extensive mechanical and design details of apparatus should not be given.

3. The abstract of the disclosure is objected to because it is not a concise statement of the technical disclosure of the patent and does not include that which is new in the art to which the invention pertains. Correction is required. See MPEP § 608.01(b).

## *Claim Rejections - 35 USC § 112*

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claims 5 and 16 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

a. The claim language in the following claims is not clearly understood:

i. As per claim 5, it is not clearly understood whether "a currently executing thread" is the same as "a currently executing thread" referred to in claim 1.

ii. As per claim 16, it is not clearly understood whether, suspending id for the second time or it is the same step claimed in claim 12. Furthermore, it is unclear whether "a currently executing thread" is the same as "a currently executing thread" referred to in claim 12.

## *Claim Rejections - 35 USC § 103*

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-3, 5-14, and 16-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ogus et al. (US 6,964,046 B1), in view of Huynh et al. (US 5,386,561) and further in view of Moore et al. (US 5,953,336).

8. As per claim 1, Ogus teaches the invention substantially as claimed including a method for scheduling thread execution, comprising:

maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which resources are allocated to a particular thread (col. 1, lines 28-61; col. 6, lines 5-27; col. 6, lines 1-6);

configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot (col. 1, lines 28-47);

maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index (col. 6, lines 2-27; col. 7, lines 1-14);

identifying a next sequential non-empty time slot that includes a queue of threads scheduled for execution during that time slot (col. 7, lines 1-20; col. 9, lines 22-36);

updating the pointer index to point to the identified next sequential non-empty time slot (col. 9, lines 22-36);

activating the thread at the top of the array of threads requesting resource allocation ( col. 10, lines 1-16).

9. Ogus doesn't explicitly teach that resources are CPU resource maintaining an array of threads requesting resource allocation based on the queue of threads, maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads;

suspending a currently executing thread upon expiration of a current timeslice associated with the currently executing thread;

calculating a next time slot during which the currently executing thread should next resume execution;

appending the suspended currently executing thread to the queue of threads scheduled for execution at the calculated time slot; appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation; and removing the thread at the top of the array of threads requesting immediate CPU resource allocation.

10. However, Huynh teaches that resources are CPU resources maintaining an array of threads requesting resource allocation based on the queue of threads and maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads (col. 5, lines 63-68; col. 6, lines 1-6);

suspending a currently executing thread upon expiration of a current timeslice associated with the currently executing thread (col. 6, lines 17-29); and

removing the thread at the top of the array of threads requesting immediate CPU resource allocation ( col. 6, lines 60-66; wherein removing the thread at the top of the array is scheduling the highest priority).

11. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Ogus and Huynh because Huynh teaching would improve scheduling techniques and system performance by giving priority to thread with higher importance and with higher demand for the CPU over lower priority thread that is not that important for the system to be scheduled.

12. The combined teaching of Ogus and Huynh doesn't explicitly teach calculating a next time slot during which the currently executing thread should next resume execution;

appending the suspended currently executing thread to the queue of threads scheduled for execution at the calculated time slot; appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation.

13. However, Moore teaches calculating a next time slot during which the currently executing thread should next resume execution;

appending the suspended currently executing thread to the queue of threads scheduled for execution at the calculated time slot; appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation (abs.; col. 4, lines 5-51; col. 8, lines 34-col. 10, lines 67).

14. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Ogus , Huynh, and Moore because Moore teaching maintaining a time slot

for the most demanded threads which improve system throughput and performance.

15. As per claim 2, the combined teaching of Ogus, Huynh, and Moore doesn't explicitly teach that each timeslice is between 10 and 100 microseconds. However, it would have been known to one of ordinary skill in the art at the time the invention was made that is well known that the time slice can have any range including 10 and 100 microseconds as claimed.

16. As per claim 3, Moore teaches the array of threads requesting immediate CPU resource allocation includes a first-in-first-out (FIFO) structure (col. 5, lines 62-65; col. 8, lines 45-67).

17. As per claim 5, Huynh teaches the step of suspending a currently executing thread includes: receiving a self-suspend request from the currently executing thread (col. 6, lines 21-25.

18. As per claim 6 , Ogus teaches advancing the index pointer by one time slot(col. 6, lines 2-27; col. 7, lines 1-14);

19. Ogus doesn't explicitly teach removing a list of any threads to be executed at the indexed time slot and appending them to the array of threads requesting immediate CPU resource allocation;

determining whether the array of threads requesting immediate CPU resource allocation is empty;

returning to the step of advancing the index pointer by one slot if it is determined that the array of threads requesting immediate CPU resource allocation does not contain any threads; and removing and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the array of threads requesting immediate CPU resource allocation is not empty.

20. However, Huynh teaches removing a list of any threads to be executed at the indexed time slot and appending them to the array of threads requesting immediate CPU resource allocation;

determining whether the array of threads requesting immediate CPU resource allocation is empty;

returning to the step of advancing the index pointer by one slot if it is determined that the array of threads requesting immediate CPU resource allocation does not contain any threads; and removing and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the array of threads requesting immediate CPU resource allocation is not empty (col. 6, lines 60-66; where activating the thread at the top of the array is scheduling/executing the highest priority thread).

22. As per claim 7, Moore teaches calculating a next time slot during which the currently executing thread should next resume execution(abs.; col. 4, lines 5-51; col. 8, lines 34- col. 10, lines 67).

21. the combined teaching of Ogus and Moore doesn't explicitly teach receiving an external event interrupt requesting CPU resource allocation for a new thread;

calculating a next time slot during which the currently executing thread should next resume execution;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation; determining a type of thread currently executing; activating the new thread if no thread is currently executing;

performing the following steps if it is determined that a non-idle thread is currently executing:

suspending the currently executing thread;

appending the currently executing thread to the end of the array of threads requesting immediate CPU resource allocation; and

activating the new thread;

performing the following steps if it is determined that an idle thread is currently executing:

suspending the currently executing thread; and

activating the new thread.

22. However, Huynh teaches appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate

CPU resource allocation(col.6, lines 1- 29; col.6, lines 60-66; wherein if the threads that need immediate attention is less priority than the executing thread, the thread will be place to the queue corresponding to its priority)

determining a type of thread currently executing (col.6, lines 1-29);

performing the following steps if it is determined that a non-idle thread is currently executing:

suspending the currently executing thread (col.6, lines 16-29);

appending the currently executing thread to the end of the array of threads requesting immediate CPU resource allocation (col.6, lines 20-21); and

activating the new thread (col.6, lines 25-30);

performing the following steps if it is determined that an idle thread is currently executing ( col.6, lines 21-25 ; wherein an idle thread is a thread that doesn't do any work that cannot run no more waiting for resource):

suspending the currently executing thread(col.6, lines 16-29); and

activating the new thread(col.6, lines 25-30).

23. As per claim 8, Ogus teaches a method for scheduling thread execution, comprising:

maintaining a plurality of circular array structures, each having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which resources are allocated to a particular thread (col. 1, lines 28-61; col. 6, lines 5-27; col. 6, lines 1-6);

configuring each time slot in each of the circular arrays to include a queue of threads scheduled for execution during that time slot (col. 1, lines 28-47);

maintaining at least one pointer index for referencing one time slot in each of the circular arrays, whereby advancement through the circular arrays is provided by advancing the pointer index (col. 6, lines 2-27; col. 7, lines 1-14);

incrementing the index pointer by one slot(col. 6, lines 2-27; col. 7, lines 1-14; col. 9, lines 23-40);

proceeding to the next array of threads requesting immediate CPU resource allocation if it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed (col. 3, lines 1-25; col. 8, lines 1-67).

24. Ogus doesn't explicitly teach that array structures associated with a plurality of discrete thread priorities, resources are CPU resources, assigning each thread to be executed a specific priority; maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays ;

removing, for each of the plurality of circular arrays, each queue of threads for the indexed time slot;

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array;

determining whether the array of threads requesting immediate resource allocation associated with a first circular array contains any threads;

proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty;

extracting its top thread if the array of threads requesting immediate CPU resource allocation contains any threads;

determining whether a priority of the top thread is greater than a priority of the currently executing thread;

calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread;

performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread: suspending the currently executing thread; activating the top thread; and

calculating the time of next execution for the suspended thread;

determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed.

25. However, Huynh teaches plurality of circular array structures associated with a plurality of discrete thread priorities (col.6, lines 1-15),

maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays (col. 5, lines 63-68; col. 6, lines 1-6);

assigning each thread to be executed a specific priority (col. 6, lines 2-6);

removing, for each of the plurality of circular arrays, each queue of threads for the indexed time slot (col. 6, lines 60-66)

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array (col. 6, lines 60-66) determining whether the array of threads requesting immediate CPU resource allocation associated with a first circular array is non-empty (col.6, lines 60-66);

proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty (fig.2; col.6, lines 25-29; col.6, lines 67 wherein, if there is no higher priority process (high priority queue is empty), there will be no preemption of the currently executing process

extracting its top thread if the array of threads requesting immediate CPU resource allocation is non-empty (fig.2; col. 6, lines 65-66);

determining whether a priority of the top thread is greater than a priority of the currently executing thread (6, lines 65-66);
performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread:

suspending the currently executing thread (col. 6, lines 17-21);

activating the top thread (fig.2, col.6, lines 60-66; wherein scheduling the execution of the highest priority thread is activating the thread that has the highest priority which is the first one on the highest priority queue); and

calculating the time of next execution for the previously executing thread (col. 6, lines 16-20);

determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed (col.6, lines 16-21; and

proceeding to the next array of threads requesting immediate CPU resource allocation if it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed (col.6, lines 1-30; wherein scheduling does depending the priority level and since the threads with the same priority are in the same queue, the next thread to be processed would be from the same priority level that need immediate attention and if all threads within the same priority have been processed, the next priority level queue is serviced).

26. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Ogus and Huynh because Huynh teaching would improve system performance and throughput.

27. The combined teaching doesn't explicitly teach calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread.

28. However, Moore teaches calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread (abs.; col. 4, lines 5-51; col. 8, lines 34- col. 10, lines 67).

29. It would have been obvious to one of ordinary skill in the art at the time the invention was made to Combine Ogus, Huynh and Moore because Moore teaching would improve scheduling techniques and system performance that will allow one to fine tune the system based

on the result of the calculation.

30. As per claim 9, the combined teaching doesn't explicitly teach that each of the plurality of circular arrays corresponds to one of four assigned priority levels: a non-real-time priority; a soft-real-time priority; a hard-real-time priority; and a critical-real-time priority.

31. However, it would have been obvious to one of ordinary skill in the art at the time the invention was made to conclude from Moore and Huynh teaching that that each of the plurality of circular arrays corresponds to one of four assigned priority levels: a non-real-time priority; a soft-real-time priority; a hard-real-time priority; and a critical-real-time priority (Moore: par.[0024], lines 12-16) which allow the system to perform all type of different priority depending on how important the execution of the task to the system.

32. As per claim 10, Moore teaches calculating a next time slot during which the currently executing thread should next resume execution(abs.; col. 4, lines 5-51; col. 8, lines 34- col. 10, lines 67).

33. The combined teaching of Ogus and Moore doesn't explicitly teach receiving an external event interrupt requesting CPU resource allocation for a new thread;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation; determining whether a priority of the new thread is greater than a priority of the currently executing thread;

appending the new thread to the end of the array of threads requesting immediate CPU resources for the associated priority if it is determined that the priority of the new thread is not greater than the priority of the currently executing thread; and

performing the following steps if it is determined that the priority of the new thread is greater than the priority of the currently executing thread: suspending the currently executing thread; calculating the time for next execution for the currently executing thread appending the currently executing thread to array associated with the calculated time slot; and activating the new thread.

34. However, Huynh teaches appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation (col.6, lines 1- 29; col.6, lines 60-66; wherein if the threads that need immediate attention is less priority than the executing thread, the thread will be place to the queue corresponding to its priority)

determining whether a priority of the new thread is greater than a priority of the currently executing thread (col. 6, lines 60-66);

appending the new thread to the end of the array of threads requesting immediate CPU resources for the associated priority if it is determined that the priority of the new thread is not greater than the priority of the currently executing thread(col.6, lines 1- 29; col.6, lines 60-66;

wherein if the threads that need immediate attention is less priority than the executing thread, the thread will be place to the queue corresponding to its priority); and

performing the following steps if it is determined that the priority of the new thread is greater than the priority of the currently executing thread:

suspending the currently executing thread (col. 6, lines 17-20);

calculating the time for next execution for the currently executing thread(col. 6, lines 16-20);

appending the currently executing thread to array associated with the calculated time slot (col.6, lines 20-21); and

activating the new thread (fig.2, col.6, lines 60-66; wherein scheduling the execution of the highest priority thread is activating the thread that has the highest priority which is the first one on the highest priority queue).

35. The combined teaching doesn't explicitly teach receiving an external event interrupt requesting CPU resource allocation for a new thread and determining whether the external event interrupt is requesting immediate CPU resource allocation. However, it would have been obvious to one of ordinary skill in the art at the time the invention was made to conclude from Moore and Huynh teaching that that interrupts are threads with priority the one with higher priority than the currently executing thread can preempt (interrupt) the currently executing thread and preemption the currently executing thread is a interrupt with higher priority than the executing thread which improve scheduling techniques and system performance by giving priority to threads with higher importance over lower priority threads that it is not that important for the system to be scheduled.

36. As per claim 11, Ogus teaches a method for scheduling thread execution, comprising:

maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread(col. 1, lines 28-61; col. 6, lines 5-27; col. 6, lines 1-6);

configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot (col. 1, lines 28-47);

maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index(col. 6, lines 2-27; col. 7, lines 1-14);

identifying a next sequential non-empty time slot containing a queue of threads scheduled for execution during that time slot (col. 7, lines 1-20; col. 9, lines 22-36);

updating the pointer index to point to the identified next sequential non-empty time slot (col. 9, lines 22-36).

37. Ogus doesn't explicitly teach maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads;

calculating a next time slot during which a currently executing thread should next resume execution;

appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot;

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;

removing the thread at the top of the array of threads requesting immediate CPU resource allocation;

determining whether the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;

maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate

CPU resource allocation is identical to the currently executing thread; suspending a currently executing thread; and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread.

38. However, Huynh teaches maintaining an array of threads requesting immediate CPU resource allocation(col. 5, lines 63-68; col. 6, lines 1-6);

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation (col. 6, lines 60-66; wherein executing the highest priority thread first the lower priority thread that were executing is appending the lower priority threads that suppose to be executing to be executing after the highest priority threads.);

determining whether the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread (col.6, lines 60-62);

removing the thread at the top of the array of threads requesting immediate CPU resource allocation (col. 6, lines 60-66; where removing the thread at the top of the array is scheduling the highest priority thread);

maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread( col. 6, lines 16-29 wherein being identical is having the same level of priority);

suspending a currently executing thread (col.6, lines 17-21); and

activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread (col. 6, lines 17-29; col. 6, lines 60-66 wherein the test of not being identical of the level of priority).

39. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Ogus and Huynh that Huynh teaching would improve scheduling techniques and system performance by giving priority to threads with higher importance over lower priority threads that it is not that important for the system to be scheduled.

40. The combined teaching of Ogus and Huynh doesn't explicitly teach calculating a next time slot during which a currently executing thread should next resume execution;

appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot.

41. However, Moore teaches calculating a next time slot during which a currently executing thread should next resume execution;

appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot(abs.; col. 4, lines 5-51; col. 8, lines 34- col. 10, lines 67).

42. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Ogus, Huynh and Moore because Moore teaching would improve the performance of important threads and also give a fair chance of lower important thread to still be scheduled and prevent CPU from being monopolized by higher priority threads.

43. As per claims 12-14, and 16-18, it is the computer-readable storage medium claims of the method claims 1-3 and 5-7 respectively. Therefore, they are rejected under the same rational.

44. As per claims 19-22, they are the computer-readable storage medium claims of the method claims 8-11 respectively. Therefore they are rejected under the same rational.

## ***Response to Arguments***

45. Applicant's arguments with respect to claims 1-3, 5-14, and 16-22 have been considered but are moot in view of the new ground(s) of rejection.

## ***Conclusion***

46. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

47. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CAROLINE ARCOS whose telephone number is (571)270-3151. The examiner can normally be reached on Monday-Thursday 7:00 AM to 5:30 PM.

48. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 571-272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

49. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/VAN H NGUYEN/
Primary Examiner, Art Unit 2194

/Caroline Arcos/
Examiner, Art Unit 2195